Architecture Overview

Version: 0.1 17/SEP/09

Introduction

This document will describe the overall approach to the architecture of Solve-o-Matic and the aims of the design. The top level aim of the system is to provide a tool which is applicable to near-real-time data reduction and analysis at the beamline, though more sophisticated (and therefore slower) functionality may be desired in the future. The context is that diffraction data has been measured recently by a user at a beamline, and that they wish to solve the structure to a sufficient extent that the success or failure of the experiment may be determined.

For each step that will be defined there may be multiple approaches and different software packages which may be used to achieve the aim. These should not, as far as possible, be hard coded and the interfaces should endeavour to remain generic. This will mean that the *process* will need to be decoupled from the implementation.

There are a number of start and end points to consider, for example it may be the case that the measurements have not yet been reduced, or that they have already been reduced but we wish to perform some difference map calculation to look for a ligand bound to an active site. In order to simplify the definitions of interfaces, four main domains will be considered:

- Data reduction, i.e. the process of taking the measured raw images and reducing them to a list of intensities or structure factor amplitudes
- Phase calculation, i.e. the calculation of phases for the list of reduced structure factor amplitudes, *via* experimental (i.e. heavy-atom) based methods or with a template structure
- Model building, i.e. the tracing of an existing electron density map with a main chain and the docking of side chains
- Model refinement, i.e. the refinement of the model from either phase calculation or model building

Clearly there will be some instances when only a subset of these domains are considered. However, the clear definition of steps will make it possible to define the intermediate interfaces and hence begin to determine the overall architecture of the system.

At this stage the implementation of sophisticated feedback between stages is not being considered. This should however be kept in mind during the composition of the interfaces to ensure that once such paths are defined they may be implemented.

Terminology

In the first instance the Solve-o-Matic system will be a set of pipelines rather than an expert system: this distinction is deliberate, as the objective is to put in place a set of tools for the user to operate quickly for feedback during the experiment.

Where more complex components are being defined design patterns will be used, as these encapsulate a substantial amount of additional meaning and will help the behaviour of the overall system, as a composition of these components, to be clearer.

As the software is already in place to perform all of the calculations that are being considered for Solve-o-Matic, no actual crystallographic software will be implemented. The system will therefore be build on a set of *underlying programs* or *software* including CCP4, PHENIX and XDS. These programs will be exposed to the system through a library of program wrappers, built on top of implementations of the *Driver* interface, as already deployed in xia2. These may in turn be composed to modules which use one or more program wrappers to perform a given task, which will satisfy a given interface definition and therefore conform to the strategy pattern. These will be finally composed within template methods (pipelines) which perform the overall task as defined above, though there may be more than one pipeline for a given task, e.g. phase calculation. In these cases a standard interface definition will be provided and the pipelines considered as strategies for the problem.

The end result of this will be that the pipelines above may be chained together and provided that the appropriate metadata is in place and the data themselves are adequate, the user should have no need to specify any direction to the system as what to actually do: it should be able to determine the most appropriate strategy from the available information. It must however remain possible for the user to specify the exact sequence of operations to perform (i.e. the strategy for each step) as they are likely to have prior knowledge or preferences which may help the overall process.

Starting Points

There are several starting points which are defined implicitly above, namely:

- Raw data
- Reduced data
- Phased reduced data, i.e. an electron density map

The level of necessary metadata associated with each of these will depend on the overall strategy used for each domain. However, from the available metadata it should be possible to establish the best high level route. Finally, it is worth noting that it may be possible to extend this process upstream, that is to include the data collection steps. This should however be kept for another day.

End Points

The main reason for limiting the end points (i.e. rather that progressing to the end of the model refinement for every data set) is to provide the necessary results as quickly as possible. However, in some cases a given step may fail: in such cases it will be necessary that the nature of the error is propogated to the user and they are able to perhaps select an alternative strategy which may succeed where the previous one failed. To this end it may be worth considering a graphical workflow editor to help will the construction of the pipelines.